42 | 软件部署实战（下）：IAM系统安全加固、水平扩缩容实战

2021-08-31 孔令飞

《Go 语言项目开发实战》 课程介绍 >

讲述：孔令飞
时长 16:04 大小 14.73M

你好，我是孔令飞。

这一讲和前面两讲，都是介绍如何基于物理机 / 虚拟机来部署 IAM 的。在前面两讲，我们了解了如何部署一个高可用的 IAM 应用，今天就再来看看 IAM 应用安全和弹性伸缩能力的构建方式。在这一讲中，我会带你加固 IAM 应用的安全性，并介绍如何具体执行扩缩容步骤。

接下来，我们先来看下如何加固 IAM 应用的安全性。

IAM 应用安全性加固

iam-apiserver、iam-authz-server、MariaDB、Redis 和 MongoDB 这些服务，都提供了绑定监听网卡的功能。我们可以将这些服务绑定到内网网卡上，从而只接收来自于内网的请求，通过这种方式，可以加固我们的系统。

我们也可以通过 iptables 来实现类似的功能，通过将安全问题统一收敛到 iptables 规则，可以使我们更容易地维护安全类设置。

这门课通过 iptables 来加固系统，使系统变得更加安全。下面，我先来对 iptables 工具进行一些简单的介绍。

iptables 简介

iptables 是 Linux 下最优秀的防火墙工具，也是 Linux 内核中 netfilter 网络子系统用户态的工具。

netfilter 提供了一系列的接口，在一个到达本机的数据包，或者经本机转发的数据包流程中添加了一些可供用户操作的点，这些点被称为 HOOK 点。通过在 HOOK 点注册数据包处理函数，可以实现数据包转发、数据包过滤、地址转换等功能。

用户通过 iptables 工具定义各种规则，这些规则通过 iptables 传给内核中的 netfilter。最终，netfilter 会根据规则对网络包进行过滤。Linux 系统一般会默认安装 iptables 软件。防火墙根据 iptables 里的规则，对收到的网络数据包进行处理。

iptables 里的数据组织结构分为表、链、规则。

表（tables）：表可以提供特定的功能，每个表里包含多个链。iptables 里面一共有 5 个表，分别是 filter、nat、mangle、raw、security。这些表，分别用来实现包过滤、网络地址转换、包重构、数据追踪处理和 SELinux 标记设置。

链（chains）：链是数据包传播的路径，每一条链中可以有一个或多个规则。当一个数据包到达一个链时，iptables 会从链中第一条规则开始，检查该数据包是否满足规则所定义的条件。如果满足，就会根据该条规则所定义的方法，处理该数据包。否则，就继续检查下一条规则。如果该数据包不符合链中任一条规则，iptables 就会根据该链预先定义的默认策略来处理数据包。

规则（rules）：规则存储在内核空间的信息包过滤表中，用来描述“如果数据包满足所描述的条件，就按照要求处理这个数据包，如果不满足，就判断下一条规则”。

其中，iptables 中表和链的种类及其功能，如下表所示：

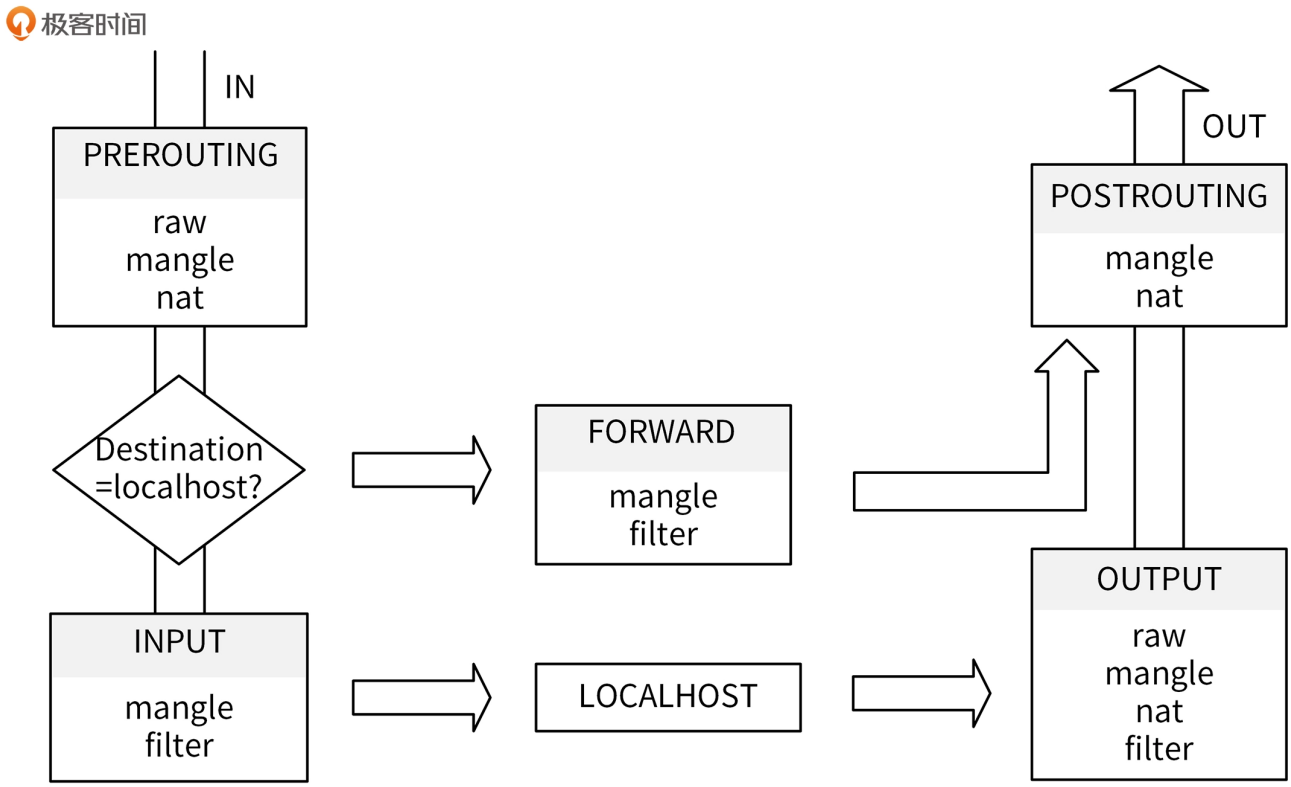
类别	名称	描述
链	PREROUTING	数据包进入路由之前，可以在此处进行DNAT
	INPUT	一般处理本地进程的数据包，目的地址为本机
	FORWARD	一般处理转发到其他机器或者network namespace的数据包
	OUTPUT	原地址为本机，向外发送，一般处理本地进程的输出数据包
	POSTROUTING	发送到网卡之前，可以在此处进行SNAT
表	FILTER	用于控制到达某条链上的数据包是继续放行、直接丢弃(drop)还是拒绝(reject)
	NAT	network address translation，网络地址转换，用于修改数据包的源地址和目的地址
	MANGLE	用于修改数据包的IP头信息
	RAW	iptables是有状态的，其对数据包有链接追踪机制，链接追踪信息在/proc/net/nf_conntrack中可以看到记录，而raw是用来去除链接追踪机制的
	SECURITY	最不常用的表，用在SELinux上

上面的表格中，五张表的处理是有顺序的。当数据包到达某一条链时，会按照 RAW、MANGLE、NAT、FILTER、SECURITY 的顺序进行处理。

到这里，我介绍了关于 iptables 的一些基础知识，但这还远远不够。要想使用 iptables 来加固你的系统，你还需要掌握 iptables 工具的使用方法。接下来，我先来介绍下 iptables 是如何处理网络数据包的。

网络数据包处理流程

网络数据包的处理流程如下图所示：



具体可以分为两个步骤。

第一步，当数据包进入网卡后，它首先进入 PREROUTING 链，根据目的 IP 判断是否转发出去。

第二步分为两种情况：如果数据包目的地是本机，它会到达 INPUT 链。到达后，任何进程都会收到它。本机上的程序可以发送数据包，这些数据包会经过 OUTPUT 链，然后经 POSTROUTING 链输出；如果数据包是要转发出去，并且内核允许转发，那么数据包会经过 FORWARD 链，最后从 POSTROUTING 链输出。

iptables 工具使用方式介绍

iptables 的功能强大，所以使用方法也非常多样。这里，我来介绍下 iptables 工具的使用方式，并给出一些使用示例。

1. 命令格式

iptables 的语法格式为：

```
1 iptables [-t 表名] 命令选项 [链名] [条件匹配] [-j 目标动作或跳转]
```

下面是一个 iptables 的使用示例：

```
1 iptables -t nat -I PREROUTING -p tcp --dport 8080 -j DNAT --to 10.0.4.88
```

这里对上面涉及到的一些参数进行说明。

表名 / 链名：指定 iptables 命令所操作的表 / 链。

命令选项：指定处理 iptables 规则的方式，例如插入、增加、删除、查看等。

条件匹配：指定对符合条件的数据包进行处理。

目标动作或跳转：防火墙处理数据包的方式。

iptables 的命令选项又分为管理控制选项和通用选项。

管理控制选项如下：

选项	说明
-A	在指定链的末尾添加（append）一条新的规则
-D	删除（delete）指定链中的某一条规则，可以按规则序号和内容删除
-I	在指定链中插入（insert）一条新的规则，默认在第一行添
-R	修改、替换（replace）指定链中的某一条规则，可以按规则序号和内容替换
-L	列出（list）指定链中所有的规则进行查看
-E	重命名用户定义的链，不改变链本身
-F	清空（flush）
-N	新建（new-chain）一条用户自己定义的规则链
-X	删除指定表中用户自定义的规则链（delete-chain）
-P	设置指定链的默认策略（policy）
-Z	将所有表的所有链的字节和数据包计数器清零
-n	使用数字形式（numeric）显示输出结果
-v	查看规则表详细信息（verbose）的信息
-V	查看版本(version)
-h	获取帮助（help）

通用选项如下：

选项	说明
–p	协议，例：iptables –A INPUT –p tcp
–s	源地址，例：iptables –A INPUT –s 192.168.1.1
–sport	源端口，例：iptables –A INPUT –p tcp ––sport 22
–dport	目的端口，例：iptables –A INPUT –p tcp ––dport 22
–i	指定入口网卡，例：iptables –A INPUT –i eth0
–o	指定出口网卡，例：iptables –A FORWARD –o eth0
–j	指定要进行的处理动作
–m	模块关键字

处理数据包的方式（目标动作或跳转）有多种，具体如下表所示：

动作	描述
ACCEPT	允许数据包通过
DROP	直接丢弃数据包，不给出任何回应信息，这时客户端会感觉请求丢失了，过了超时时间才会有反应
REJECT	拒绝数据包通过，必要时会给数据发送端一个响应信息，客户端刚请求就会收到拒绝的信息
LOG	在日志文件/var/log/messages中记录日志信息，然后将数据包传递给下一条规则
QUEUE	防火墙将数据包移交到用户空间
RETURN	防火墙停止执行当前链中的后续Rules，并返回到调用链
REDIRECT	端口重定向，在本机做端口映射
MARK	做防火墙标记
DNAT	目标地址转换
SNAT	源地址转换，解决内网用户用同一个公网地址上网的问题
MASQUERADE	地址伪装，是SNAT的一种特殊形式，适用于动态的、临时会变的IP上

上面，我介绍了 iptables 工具的使用方式。因为内容有点多，你可能仍然不知道如何使用 iptables 工具。没关系，接下来你可以结合我举的一些例子来看下。

2. 命令示例

下面的命令示例，默认使用了 FILTER 表，也即规则存放在 FILTER 表中，相当于每一条 iptables 命令都添加了`-t filter`参数。

1. 拒绝进入防火墙的所有 ICMP 协议数据包：

```
$ iptables -I INPUT -p icmp -j REJECT
```

2. 允许防火墙转发除 ICMP 协议以外的所有数据包：

```
$ iptables -A FORWARD -p ! icmp -j ACCEPT
```

3. 拒绝转发来自 192.168.1.10 主机的数据，允许转发来自 192.168.0.0/24 网段的数据：

```
$ iptables -A FORWARD -s 192.168.1.11 -j REJECT
$ iptables -A FORWARD -s 192.168.0.0/24 -j ACCEPT
```

4. 丢弃从外网接口（eth1）进入防火墙本机的源地址为私网地址的数据包：

```
$ iptables -A INPUT -i eth1 -s 192.168.0.0/16 -j DROP
$ iptables -A INPUT -i eth1 -s 172.16.0.0/12 -j DROP
$ iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j DROP
```

5. 只允许管理员从 202.13.0.0/16 网段使用 SSH 远程登录防火墙主机：

复制代码

```
$ iptables -A INPUT -p tcp --dport 22 -s 202.13.0.0/16 -j ACCEPT
$ iptables -A INPUT -p tcp --dport 22 -j DROP
```

6. 允许本机开放从 TCP 端口 20-1024 提供的应用服务：

复制代码

```
$ iptables -A INPUT -p tcp --dport 20:1024 -j ACCEPT
$ iptables -A OUTPUT -p tcp --sport 20:1024 -j ACCEPT
```

7. 允许转发来自 192.168.0.0/24 局域网段的 DNS 解析请求数据包：

复制代码

```
$ iptables -A FORWARD -s 192.168.0.0/24 -p udp --dport 53 -j ACCEPT
$ iptables -A FORWARD -d 192.168.0.0/24 -p udp --sport 53 -j ACCEPT
```

8. 禁止其他主机 ping 防火墙主机，但是允许从防火墙上 ping 其他主机：

复制代码

```
$ iptables -I INPUT -p icmp --icmp-type Echo-Request -j DROP
$ iptables -I INPUT -p icmp --icmp-type Echo-Reply -j ACCEPT
$ iptables -I INPUT -p icmp --icmp-type destination-Unreachable -j ACCEPT
```

9. 禁止转发来自 MAC 地址为 00：0C：29：27：55：3F 的数据包和主机的数据包：

复制代码

```
$ iptables -A FORWARD -m mac --mac-source 00:0c:29:27:55:3F -j DROP
```

10. 对外开放 TCP 端口 20、21、25、110，以及被动模式 FTP 端口 1250-1280：

复制代码

```
$ iptables -A INPUT -p tcp -m multiport --dport 20,21,25,110,1250:1280 -j ACCE
```

11. 禁止转发源 IP 地址为 192.168.1.20-192.168.1.99 的 TCP 数据包：

```
$ iptables -A FORWARD -p tcp -m iprange --src-range 192.168.1.20-192.168.1.99
```

12. 禁止转发与正常 TCP 连接无关的非 syn 请求数据包：

```
$ iptables -A FORWARD -m state --state NEW -p tcp ! --syn -j DROP
```

13. 拒绝访问防火墙的新数据包，但允许响应连接或与已有连接相关的数据包：

```
$ iptables -A INPUT -p tcp -m state --state NEW -j DROP
$ iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
```

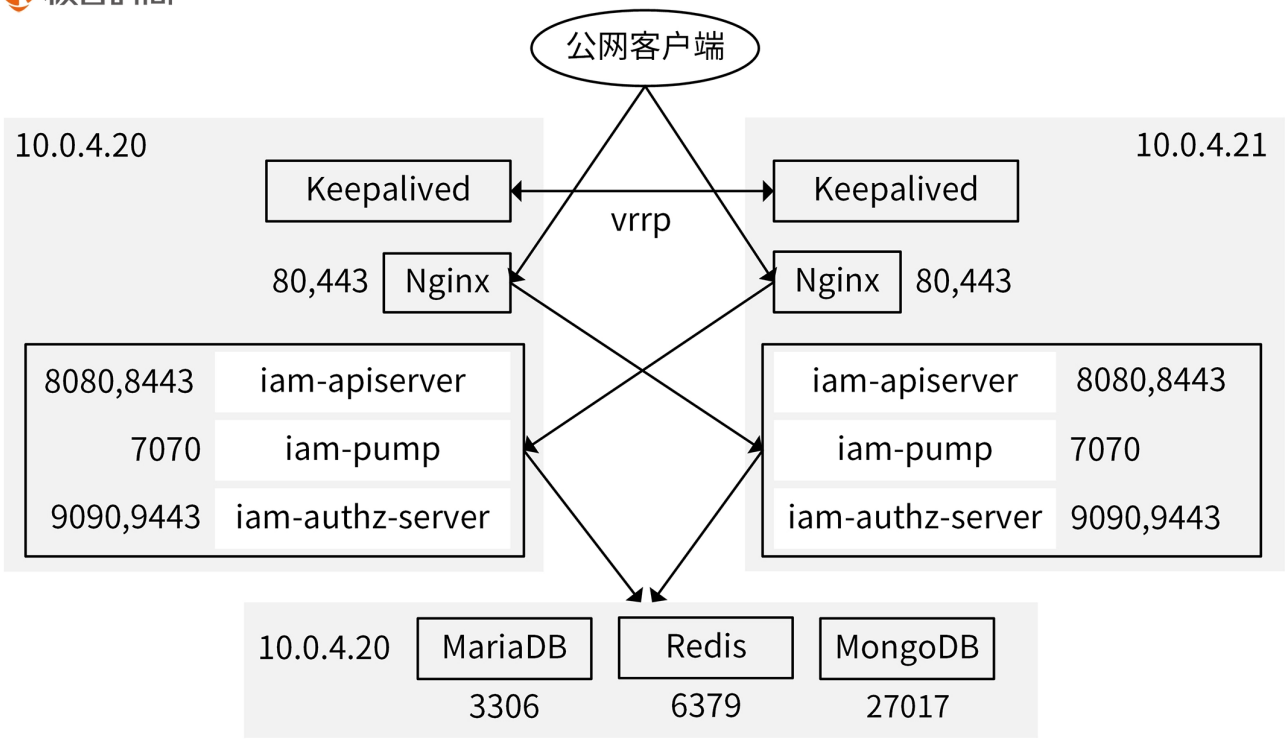
14. 只开放本机的 web 服务（80）、FTP(20、21、20450-20480)，放行外部主机发往服务器其他端口的应答数据包，将其他入站数据包都进行丢弃处理：

```
$ iptables -I INPUT -p tcp -m multiport --dport 20,21,80 -j ACCEPT
$ iptables -I INPUT -p tcp --dport 20450:20480 -j ACCEPT
$ iptables -I INPUT -p tcp -m state --state ESTABLISHED -j ACCEPT
$ iptables -P INPUT DROP
```

到这里，我们已经了解了 iptables 的功能，下面来看看如何使用 iptables 来加固 IAM 应用。我把它分成内网不安全和内网安全两种情况。

IAM 安全加固（内网不安全）

在设置 iptables 规则之前，我们需要先梳理系统的访问关系，然后根据这些访问关系设置 iptables 规则。访问关系如下图所示：



你可以看到，IAM 系统服务互访关系分为下面这 4 种：

允许公网客户端访问 Nginx 的 80 和 443 端口。

Keepalived 服务之间能够互发 VRRP 协议包。

Nginx 访问各节点上 iam-apiserver、iam-authz-server 和 iam-pump 组件开启的 HTTP/HTTPS/GRPC 服务。

iam 服务可以从各节点访问 Redis、MariaDB、MongoDB 数据库。

这里，我们假定 IAM 系统部署在一个非常大的内网中，该内网部署了很多其他团队的服务，有很多其他团队的研发、测试等人员在内网中执行各种操作。也就是说，我们处在一个不安全的内网中。这时候，如果要加固我们的系统，最安全的方式是屏蔽掉未知的来源 IP。

内网不安全的情况下，加固系统可以分为 3 大步骤，每个步骤中又有一些小步骤。另外，需要新增节点或者删除节点时，也需要进行一些变更操作。下面我们来具体看下。

第一步，设置防火墙规则。

基于上面说到的几种互访关系，我们可以在各个节点上设置 iptables 规则来加固系统。我将这些规则设置编写成了 go 工具，用来自动生成设置这些规则的 shell 脚本。

具体设置的过程可以分为 5 步。

1. 进入 iam 项目源码根目录。
2. 配置 accesss.yaml（工具根据此配置，自动生成 iptables 设置脚本），内容如下（位于configs/access.yaml文件）：

复制代码

```
# 允许登录SSH节点的来源IP，可以是固定IP(例如10.0.4.2)，也可以是个网段，0.0.0.0/0代表不限
ssh-source: 10.0.4.0/24

# IAM应用节点列表（来源IP）
hosts:
  - 10.0.4.20
  - 10.0.4.21

# 来源IP可以访问的应用端口列表（iam-apiserver, iam-authz-server, iam-pump对外暴露的的
ports:
  - 8080
  - 8443
  - 9090
  - 9443
  - 7070

# 来源IP可以访问的数据库端口列表（Redis, MariaDB, MongoDB）
dbports:
  - 3306
  - 6379
  - 27017
```

上面的配置中，我们指定了允许登陆机器的子网、Nginx 需要访问的端口列表和各节点需要访问的数据库端口列表。

3. 生成 iptables 初始化脚本：

复制代码

```
$ go run tools/geniptables/main.go -c access.yaml -t app -a -o firewall.sh
$ ls firewall.sh
firewall.sh
```

你可以打开 firewall.sh 文件，查看该脚本设置的规则。

4. 将 firewall.sh 脚本拷贝到 10.0.4.20 和 10.0.4.21 节点执行：

```
$ scp firewall.sh root@10.0.4.20:/tmp/
$ scp firewall.sh root@10.0.4.21:/tmp/
```

登陆 10.0.4.20 和 10.0.4.21 机器，执行`/tmp/firewall.sh`。

5. 在 10.0.4.20（数据库节点）节点上，设置 iptables 规则，以允许各节点访问：

因为数据库节点也位于 10.0.4.20 节点，所以只需要添加新的 rule，并将`iptables -A INPUT -j DROP`规则放到最后执行即可。

```
$ go run tools/geniptables/main.go -c access.yaml -t db -o addrules.sh
```

然后，将 addrules.sh 脚本拷贝到 10.0.4.20 节点执行。

注意，因为 iptables 是按顺序进行规则过滤的，所以需要将`iptables -A INPUT -j DROP`规则放在新设置规则的后面，否则执行不到新设置的规则。你可以在设置完 iptables 规则之后，执行下面的命令来将 DROP 放到最后：

```
iptables -A INPUT -j LOG --log-level 7 --log-prefix "Default Deny"
iptables -A INPUT -j DROP
```

生成的 addrules.sh 脚本加入以上设置。

第二步，设置重启自动加载 iptables 规则。

前面我们在各个节点设置了 iptables 规则，但是这些规则在系统重启后会丢失。为了使系统重启后自动重新设置这些规则，我们需要将当前的 iptables 规则保存起来，让系统重启时自动加载。需要进行下面两个步骤。

1. 保存现有的规则：

```
$ sudo iptables-save > /etc/sysconfig/iptables
```

复制代码

2. 添加下面的命令行到 /etc/rc.d/rc.local 文件中：

```
$ iptables-restore < /etc/sysconfig/iptables
```

复制代码

第三步，自动化。

在上面的步骤中，我们自动生成了 iptables 规则，并手动登陆到节点进行设置。你肯定也发现了，整个流程手动操作过多，容易出错，效率还低。你可以参考设置过程，将这些设置工作自动化，比如编写脚本，一键刷新所有节点的 iptables 规则。

另外，我们再来看下在新增节点和删除节点两种场景下，如何设置 iptables 规则。

场景 1：新增节点

如果我们要扩容一个节点，也需要在新节点设置防火墙规则，并在数据库节点设置防火墙规则允许来自新节点的访问。

假如我们新增一个 10.0.4.22 节点，这里要设置防火墙规则，需要下面的 4 个步骤。

1. 编辑 access.yaml，在 hosts 列表下新增 10.0.4.22 节点 IP。编辑后内容如下：

```
# 允许登录SSH节点的来源IP，可以是固定IP(例如10.0.4.2)，也可以是个网段，0.0.0.0/0代表不限
ssh-source: 10.0.4.0/24
```

复制代码

```
# IAM应用节点列表（来源IP）
hosts:
  - 10.0.4.20
  - 10.0.4.21
  - 10.0.4.22

# 来源IP可以访问的应用端口列表（iam-apiserver, iam-authz-server, iam-pump对外暴露的端
ports:
  - 8080
  - 8443
  - 9090
  - 9443
  - 7070

# 来源IP可以访问的数据库端口列表（Redis, MariaDB, MongoDB）
dbports:
  - 3306
  - 6379
  - 27017
```

2. 在 10.0.4.22 节点设置 iptables 规则：

复制代码

```
$ go run tools/geniptables/main.go -c access.yaml -t app -a -o firewall.sh
```

将 firewall.sh 脚本拷贝到 10.0.4.22 节点，并执行。

3. 在已有节点新增规则，允许来自 10.0.4.22 的 Nginx 服务的访问：

复制代码

```
$ go run tools/geniptables/main.go -c access.yaml -t app 10.0.4.22 -o addrules
```

将 addrules.sh 脚本拷贝到存量节点，并执行。

4. 在数据库节点新增 iptables 规则，以允许来自新节点的访问：

复制代码

```
$ go run tools/geniptables/main.go -c access.yaml -t db 10.0.4.22 -o addrules.
```

将 addrules.sh 脚本拷贝到 10.0.4.20 节点执行即可。

场景 2：删除节点。

如果我们要删除一个节点，需要在保留的节点和数据库节点中，将该节点的访问权限删除。假如我们要删除 10.0.4.22 节点，设置防火墙规则需要下面 3 个步骤。

1. 在保留节点删除 10.0.4.22 节点访问权限：

```
$ go run tools/geniptables/main.go -c access.yaml -t app --delete 10.0.4.22 -o
```

复制代码

将 delete.sh 脚本拷贝到保留节点（10.0.4.20，10.0.4.21），并执行。

2. 在数据库节点删除 10.0.4.22 节点访问权限：

```
$ go run tools/geniptables/main.go -c access.yaml -t db --delete 10.0.4.22 -o
```

复制代码

将 delete.sh 脚本拷贝到 10.0.4.20 节点执行即可。

3. 将下线的节点从 access.yaml 文件中的 hosts 部分删除。

IAM 安全加固（内网安全）

这里，我们来看第二种情况：假定我们系统部署在一个安全的内网环境中，这时候加固系统就会变得异常简单，只需要允许来源 IP 为内网 IP 的客户端访问我们提供的各类端口即可。在我们设置完 iptables 规则之后，后续再新增或者删除节点，就不需要再做变更了。

具体可以分为 5 个步骤。

第一步，进入 iam 项目源码根目录。

第二步，配置 accesss.yaml（工具根据此配置，自动生成 iptables 设置脚本），内容如下（configs/access.yaml文件）：

```
# 允许登录SSH节点的来源IP，可以是固定IP(例如10.0.4.2)，也可以是个网段，0.0.0.0/0代表不限
ssh-source: 10.0.4.0/24

# 来源IP可以访问的应用端口列表（iam-apiserver, iam-authz-server, iam-pump对外暴露的端
ports:
  - 8080
  - 8443
  - 9090
  - 9443
  - 7070

# 来源IP可以访问的数据库端口列表（Redis, MariaDB, MongoDB）
dbports:
  - 3306
  - 6379
  - 27017
```

上面配置中，我们仅仅指定了 IAM 服务端口和数据库端口。

第三步，生成 iptables 初始化脚本：

```
$ go run tools/geniptables/main.go -c access.yaml -t app --cidr=10.0.4.0/24 -a
$ ls firewall.sh
firewall.sh
```

第四步，将 firewall.sh 脚本拷贝到 10.0.4.20 和 10.0.4.21 节点执行：

```
$ scp firewall.sh root@10.0.4.20:/tmp/
$ scp firewall.sh root@10.0.4.21:/tmp/
```

登陆 10.0.4.20 和 10.0.4.21 机器执行 `/tmp/firewall.sh`。

第五步，在 10.0.4.20（数据库节点）节点上，设置 iptables 规则，以允许各节点访问。

因为数据库节点也位于 10.0.4.20 节点，所以只需要添加新的 rule，并将 `iptables -A INPUT -j DROP` 规则放到最后执行即可。

```
$ go run tools/geniptables/main.go -c access.yaml -t db --cidr=10.0.4.0/24 -o
```

复制代码

然后，将 `addrules.sh` 脚本拷贝到 10.0.4.20 节点执行。

如果要增加节点，你只需要重新执行第三步，生成 firewall.sh 脚本，并将 firewall.sh 脚本拷贝到新节点上执行即可。删除节点，则不需要做任何操作。

接下来，我们再来看下如何对 IAM 应用进行弹性伸缩操作。

弹性伸缩

弹性伸缩包括扩容和缩容。扩容是指当业务量越来越大时，能够很容易地增加计算节点，来分散工作负载，从而实现计算等能力的扩展。缩容是指当业务量变小时，能够很容易地减少计算节点，从而减小成本。

在系统上线初期，通常业务量不会很大，但是随着产品的迭代，用户量的增多，系统承载的请求量会越来越多，系统承载的压力也会越来越大。这时，就需要我们的系统架构有能力进行水平扩容，以满足业务需求，同时避免因为系统负载过高造成系统雪崩。

一些电商系统，在双 11 这类促销活动之前会提前扩容计算节点，以应对即将到来的流量高峰。但是活动过后，流量会逐渐下降，这时就需要我们的系统有能力进行缩容，以减少计算节点，从而节省成本。

一个可伸缩的系统架构，是我们在进行系统设计时必须要保证的。如果系统不具有伸缩性，那么当我们后期需要扩缩容时，就需要对代码进行大改，不仅会增加额外的工作量，还会拖累产品的迭代速度。而且你想想，改完之后还要测试，发布之后，还可能因为代码变更引入 Bug。总之，不具伸缩性的系统架构可以说是后患无穷。

IAM 系统在设计之初就考虑到了系统的伸缩能力，我们可以很容易地对系统进行扩缩容。下面，我来分别介绍下如何对系统进行扩容和缩容。

系统扩容

系统扩容的步骤很简单，你只需要进行下面这 5 步：

1. 根据需要申请计算节点，如无特殊需求，计算节点的配置、操作系统等要跟已有的节点保持一致。
2. 在新的节点上部署 iam-apiserver、iam-authz-server、iam-pump，部署方式跟部署其他节点一样。
3. 在新节点部署 Nginx，并将新节点的 IP 加入到已有所有节点的 Nginx upstream 配置中，重启 Nginx。
4. 在新节点部署 Keepalived，并将新节点的 IP 加入到已有所有节点的 unicast_peer 配置中，重启 Keepalived。
5. 修改 iptables 规则，并刷新所有机器的 iptables。

系统缩容

系统缩容是系统扩容的逆向操作，也是 5 个步骤：

1. 根据需要，确定要删除的节点。
2. 关闭待删除节点的 iam-apiserver、iam-authz-server、iam-pump 服务。
3. 从所有保留节点的 Nginx upstream 配置中，删除待删除节点的 IP 地址, 重启 Nginx。
4. 从所有保留节点的 Keepalived unicast_peer 配置中，删除待删除节点的 IP 地址, 重启 Keepalived。
5. 修改 iptables 规则，并刷新所有保留机器的 iptables。

总结

安全对于应用软件来说至关重要，在部署应用时，也一定要评估应用的安全性，并采取一定的措施来保证安全性。

在进行软件部署时，保证应用安全性最简单有效的方式是使用 iptables 规则来加固系统。实现思路也很简单，就是使用 iptables 规则，只允许特定来源的 IP 访问特定的端口。

在业务正式上线之后，可能会遇到业务高峰期或低峰期。业务高峰期，可能需要添加机器，提高系统的吞吐量，可以在新机器上安装需要扩容的服务组件，并安装和配置好 Nginx 和 Keepalived，之后将该服务器添加到 Nginx 的 upstream 中。在业务低峰期时，可以将服务器从 Nginx 的 upstream 列表中移除，并关停 IAM 应用的服务。

课后练习

1. 请根据这一讲学习的内容，再增扩容一台机器。
2. 思考下，你在应用部署时，还有哪些比较好的应用安全加固方法，欢迎在留言区分享。

欢迎你在留言区与我交流讨论，我们下一讲见。

分享给需要的人，Ta订阅后你可得 24 元现金奖励

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 41 | 软件部署实战（中）：IAM 系统生产环境部署实战

下一篇 43 | 技术演进（上）：虚拟化技术演进之路

更多学习推荐

精选留言 (3)

写留言

hi,guy
2021-08-31

后面能不能补充下项目的可观测性章节，如此多服务怎么治理，有什么好的方法没有？

作者回复: 可观测性、多服务治理这属于云原生了，这个专栏会涉及但不多。

1 2

惜心（伟祺）
2021-09-03

这课值了 整个大工程开发的全套啊

展开

作者回复: 哈哈哈，学完，能够完整开发一个Go项目了

daz2yy
2021-09-01

系统缩容的时候逆向操作应该是从修改 iptables 开始 => keepalived => Nginx => 关闭

服务 => 删除节点；不然先关闭了服务，这时候 nginx 还是会把流量导过来的吧？

作者回复: 你这种方法更好，我优化下，强！